Enhanced Yap

I love Yap.
But I have gotten used to some of the Emacs bindings, and I'm always wanting to just execute some Postscript code without having to copy it, find Yap, launch it, paste it in... blah blah blah.
Since the Yap source is available... I made some small changes...

- added the EmacsText object from the MiniExample TextORama (available from ftp.next.com:/pub/NeXTanswers
- added a Services command, so you can open the selection in another program in Yap easily
- added some 'file types' to the attributes, so you have the option of .eps .ps or .font files opening in Yap instead of some other program.

There is still lots I'd like to do, but this is how far I've gotten.   Time is a very precious resource.
- be able to print the YapOutput window
- be able to save the YapOutput window as a TIFF file
- step though the Postscript code one line at a time
- display the stack, any output values from the Postscript

However, if you have done any of this, or want to do any of it, feel free to send me your changes!   I'll include them with mine.

Scott Anguish
sanguish@digifix.com
Digital Fix Development

Yap

Yap, *Yet Another Previewer*, is a simple text-editor which provides the ability to interactively execute the documents as PostScript programs. Yap demonstrates:

· Execution of random PostScript code in a separate window server context
· Caching drawing for fast scrolling purposes
· Multi-window documents loaded from nib files containing ScrollViews with subclass of Text
· Pasting PostScript as text
· A simple Find panel

`YapOutput.m` and `YapWrap.psw` contain the code which creates a separate context that is focused on a window belonging to the application's main context and send PostScript down to this second context for execution. This second context is kept around as long as no errors occur; however, in case of errors a new one is created.

`YapDocument.m` shows the YapDocument class which manages the documents. The source code for YapDocument demonstrates, among other things, how to read a `.nib` file over and over again to create multiple document windows.

If your application needs to import EPS files, then its probably best to stick to the `NXEPSImageRep` class (or `NXImage`, which uses `NXEPSImageRep` to manage its EPS representations) rather than using the code in `YapOutput.m`. `NXEPSImageRep` uses a separate context to execute EPS files in a manner similar to Yap.